МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Курганский государственный университет»

Кафедра информационных технологий и методики преподавания информатики

ТЕОРИЯ АЛГОРИТМОВ

Методические рекомендации
для проведения лабораторных работ
для студентов очной и заочной форм обучения направления 230700.62
«Прикладная информатика»

Курган 2013

Кафедра: «Информационные технологии и методика преподавания информатики»
Дисциплины: «Теория алгоритмов»
(направление 230700.62)
«Теоретические основы информатики»,
(направление 230700.62).

Составила: канд. пед. наук Т.А. Никифорова

Утверждены на заседании кафедры «18» ноября 2013 г.
Рекомендованы методическим советом университета «19» ноября 2013 г.

Лабораторная работа №1. **АЛГОРИМЫ. СВОЙСТВА АЛГОРИТМОВ**

Краткие сведения

1.1 Различные подходы к понятию «алгоритм»

Алгоритм – это фундаментальное понятие математики и программирования. Слово «алгоритм» возникло в Европе после перевода на латынь книги Абдуллы (Абу Джафара) Мухаммеда бен Муса аль-Хорезми (из города Хорезма), написанной в 825 году н.э., в которой были описаны способы выполнения арифметических действий над многозначными числами. Аль-Хорезми тогда писалось как «алгоритми». Первым дошедшим до нас алгоритмом в его интуитивном понимании – конечной последовательности элементарных действий, решающих поставленную задачу – считается предложенный Евклидом в III веке до н.э. алгоритм нахождения наибольшего общего делителя двух чисел (алгоритм Евклида). Вплоть до начала XX века само слово «алгоритм» употреблялось в устойчивом сочетании «алгоритм Евклида». Для описания пошагового решения других математических задач использовалось слово «метод». Рассмотрим различные подходы к понятию.

В соответствии с ГОСТ 19.004-80 «***алгоритм*** – это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату». Также под ***алгоритмом*** понимают конечный набор правил (процедур или команд), однозначно раскрывающих содержание и последовательность выполнения операций для систематического решения определенного класса задач за конечное число шагов.

Определение А.Н. Колмогорова: ***Алгоритм*** – это всякая система вычислений, выполняемых по строго определенным правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи.

Определение А.А. Маркова: ***Алгоритм*** – это точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.

Основные свойства алгоритма: детерминированность, массовость, результативность, дискретность, а также наличие входных и выходных данных.

Детерминированность – определенность, однозначность действий в каждом возможном случае, определяет однозначность результата работы алгоритма при заданных исходных данных.

Массовость – возможность варьирования исходных данных в определенных пределах, определяет пригодность использования алгоритма для решения множества задач данного класса.

Результативность алгоритма означает, что для любых допустимых исходных данных он должен через конечное число шагов (или итераций) завершить свою работу.

Дискретность – возможность разбиения определенного алгоритмического процесса на отдельные элементарные этапы, возможность реализации которых человеком или ЭВМ не вызывает сомнения, а результат выполнения каждого элементарного этапа вполне определен и понятен.

Определенность – однозначная определенность результатов выполнения каждого шага алгоритма.

Конечность – однозначная определенность результатов выполнения каждого шага алгоритма за конечное время.

Формальность – алгоритм не должен допускать неоднозначности толкования действий для исполнителя.

1.2 Словесное представление алгоритмов

Пример 1.1 Пусть, например, требуется найти сумму S первых n элементов одномерного числового массива X, т.е $\boldsymbol{S = \sum_{i=1}^{n} x_i}$.

В этом случае описание алгоритма может быть представлено словесно как последовательность шагов. На каждом шаге обычно выполняется некоторое действие (оператор присваивания) или осуществляется проверка некоторого условия (условный переход) (рисунок 1.1).

Шаг 1. S := 0.
Шаг 2. i: = 1.
Шаг 3. S := S + x_i.
Шаг 4. i: = i + 1.
Шаг 5. i ≤ n ?
Если да, идти к шагу 3,
иначе — к шагу 6.
Шаг 6. Конец.

Рисунок 1.1 – Словесное описание алгоритма

1.3 Средства графического представления алгоритмов. Язык блок-схем

Схемой называется графическое изображение логической структуры алгоритма, в котором каждый этап процесса обработки информации представляется в виде геометрических фигур, конфигурация которых определяет характер обозначаемых действий. Условные графические изображения, используемые при построении схем, называются *символами*.

Система символов и правила построения алгоритмов определены соответствующими стандартами (рисунок 1.2).

Рисунок 1.2 – Основные символы схем алгоритмов

Символы схемы располагаются сверху вниз. Линии соединения символов – линии потока, показывают направление процесса обработки. Стрелки на соединяющих линиях не ставят при направлениях сверху-вниз и слева-направо; противоположные направления показывают стрелкой на линии потока. Линия потока может изображаться как вертикально, так и горизонтально. За исключением символа «решение» остальные символы, относящиеся к процессу

обработки информации, имеют один вход и один выход линий потока. В символ «решение» линии потока входят вертикально только один раз, а выходить могут как вертикально, так и горизонтально.

Пример 1.2 Пусть требуется определить время встречи двух пешеходов, идущих навстречу друг другу.

Решение:

1 Содержательная постановка задачи

Искомой величиной является время движения пешеходов t. Исходными данными должны быть начальное расстояние L и скорости движения пешеходов v_1 и v_2.

Далее уточним: как движутся пешеходы. Будем считать, что они движутся с постоянными скоростями и вышли на встречу друг другу одновременно.

Точная постановка задачи приобретает вид: «Два пешехода на расстоянии L одновременно начали движение со скоростью v_1 (первый) и v_2 (второй) навстречу друг другу. Найти время движения двух пешеходов до места встречи».

Дано: L – начальное расстояние ($L \in R$);

v_1 – скорость первого пешехода ($v_1 \in R$);

v_2 – скорость второго пешехода ($v_2 \in R$).

Требуется *найти*: – время движения до встречи ($t \in R$).

2 Построение математической модели

Пусть S_1 и S_2 – путь, пройденный соответственно первым и вторым пешеходом до встречи, тогда движение этих пешеходов можно описать следующей системой:

$$\begin{cases} L = S_1 + S_2; \\ S_1 = v_1 \cdot t; \\ S_2 = v_2 \cdot t. \end{cases}$$ При условии, что $$\begin{cases} L>0; \\ v_1>0; \\ v_2>0. \end{cases}$$ Это условия допустимости данных.

Метод решения этой задачи получается решением системы уравнений.

$L = v_1 t + v_2 t;$

$L = t\ (v_1 + v_2);$

$t = L/(v_1 + v_2);$ Эта формула выражает метод исследования.

3 Алгоритм моделирования

Решение этой задачи можно организовать в виде диалога, в котором программа предлагает ввести все исходные данные, проверяет данные на допустимость (если данные недопустимы, выдает сообщение о недопустимых данных), при правильном вводе данных рассчитывает время до встречи и выводит на экран соответствующее сообщение.

Рассмотрим один из сценариев решения этой задачи:

Алг определение времени в пути
Нач
Вывод (‘движение пешеходов’)
Запрос (‘расстояние L=’; L)
Запрос (‘скорость первого v_1=’; v_1)
Запрос (‘скорость второго v_2=’; v_2)
Если L<=0

То вывод(‘недопустимое расстояние’)
Иначе если $v_1<=0$ **и** $v_2<=0$
то вывод(‘недопустимые скорости’)
иначе $t := L/(v_1 + v_2)$
вывод(‘время пути t=’; t)
конесли
конесли
кон

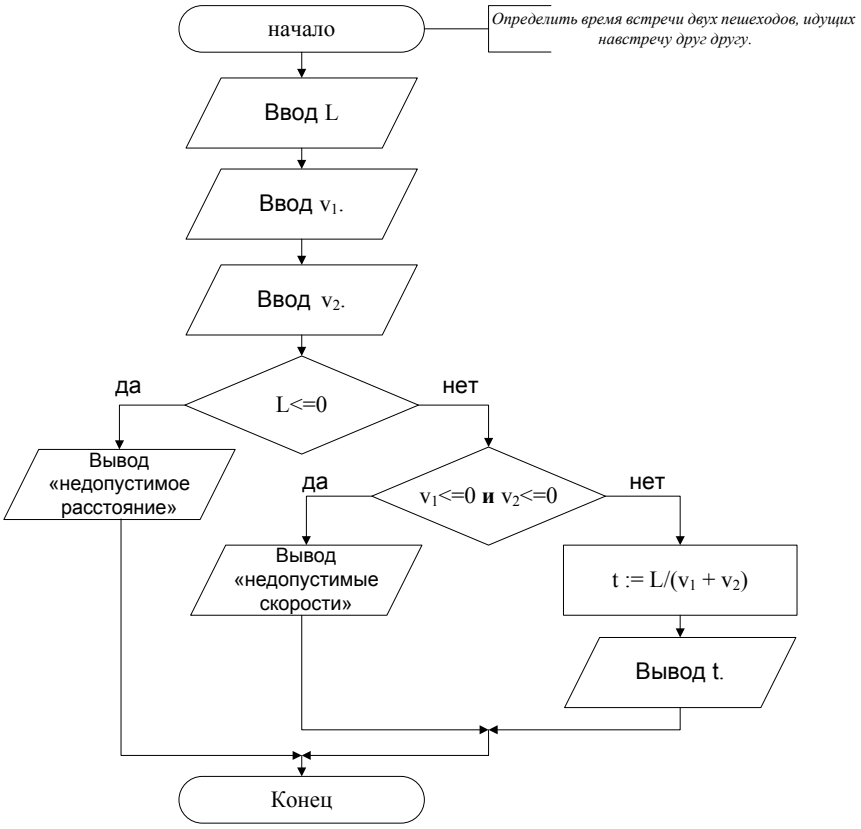


Рисунок 1.3 – Блок-схема

Замечание 1 Возможен другой алгоритм решения задачи нахождения времени встречи двух пешеходов.

Замечание 2 Для построения блок-схем можно использовать программу Microsoft Visio Professional (Файл→Создать→Блок-схема→Простая блок-схема).

1.4 Структурограммы

Другим средством графического изображения процесса передачи управления (исполнения) являются так называемые *структурограммы*. По имени их авторов такой способ изображения называют еще диаграммами или схемами Насси–Шнейдермана.

Структурограмма (схема Насси-Шнейдермана) – это схема, иллюстрирующая структуру передачи управления внутри модуля с помощью вложенных друг в друга блоков. Способ представления модуля программы с помощью схем Насси-Шнейдермана позволяет изображать передачу управления от блока к блоку не с помощью явного указания линий перехода, а с помощью вложенных структур. Каждый блок имеет форму прямоугольника и может быть вписан в любой внутренний прямоугольник любого другого блока. Запись внутри блока производится на естественном языке или с помощью математических обозначений.

Основные символы структурограмм и их связь с аналогичными конструкциями псевдокода показаны ниже.

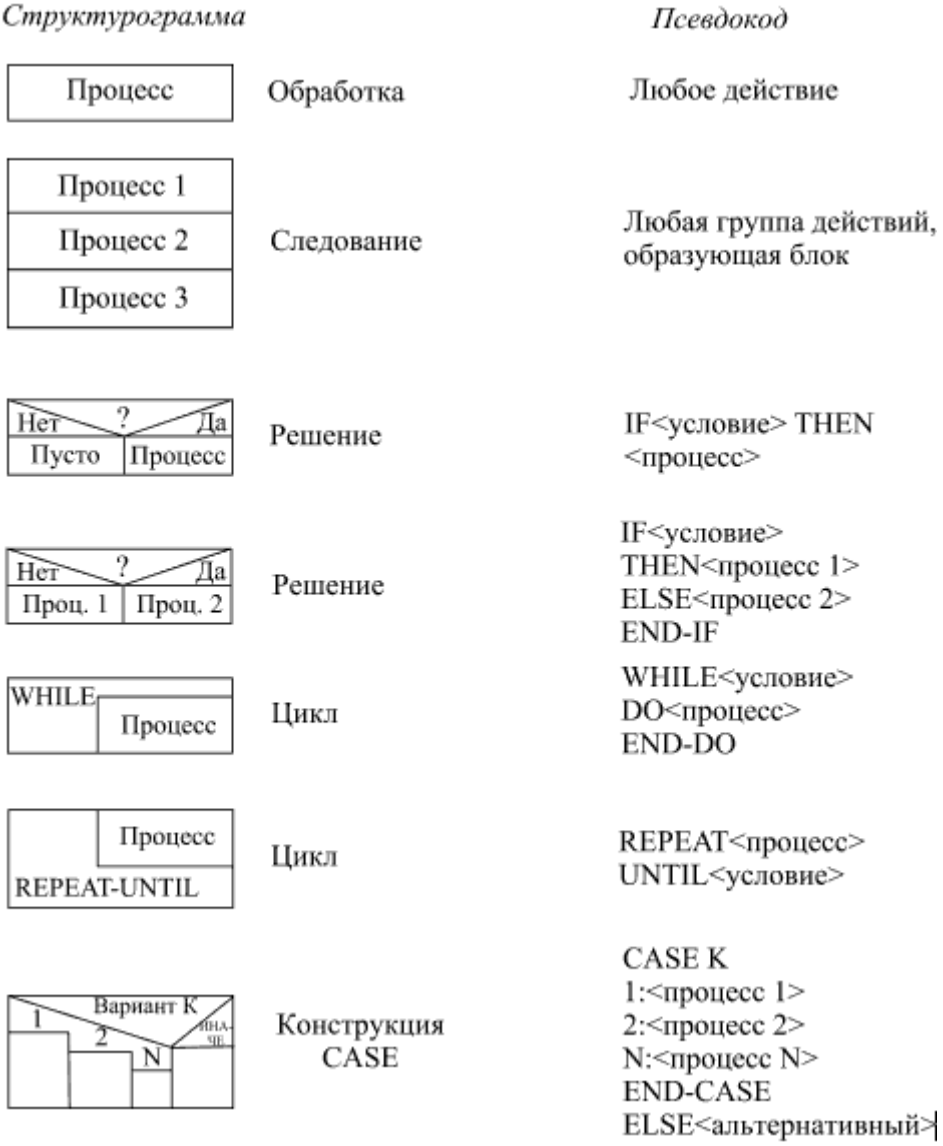


Рисунок 1.4 – Основные символы структурограмм и их связь с аналогичными конструкциями псевдокода

Рассмотрим действия, описываемые символами структурограмм.

Символ «*Следование*» объединяет ряд следующих друг за другом процессов обработки. В отдельные прямоугольники записываются логически завершенные шаги программы. Управление начинает свой путь на внешней стороне верхнего прямоугольника, проходит через каждый прямоугольник и завершает путь на внешней стороне нижнего прямоугольника.

Символ «*Решение*» применяется для обозначения конструкций IF…THEN…ELSE... Условие (вопрос) располагается в верхнем треугольнике, варианты ответов – по его сторонам, а процессы обработки обозначаются прямоугольниками.

Символ CASE представляет собой множественное ветвление.

1.5 Алгоритм Евклида нахождения наибольшего общего делителя двух чисел

Наибольшим общим делителем (НОД) двух целых чисел называется такое наибольшее по модулю число, которое делит эти два числа. Идея метода

проста. Из набора выбираются любые два ненулевых числа, и большее из них (или любое, если числа равны) заменяется разностью этих чисел. Этот процесс повторяется до тех пор, пока не останется одно ненулевое число. Это число и будет наибольшим общим делителем исходного набора, состоящего из натуральных чисел.

Пример 1.3 Словесное описание алгоритма Евклида нахождения НОД двух чисел делением.

Решение

1 Большее число делим на меньшее.

2 Если делится без остатка, то меньшее число и есть НОД (следует выйти из цикла).

3 Если есть остаток, то большее число заменяем на остаток от деления.

4 Переходим к пункту 1.

Пример 1.4 Описание алгоритма Евклида блок-схемой.

Решение (см. блок-схему справа).

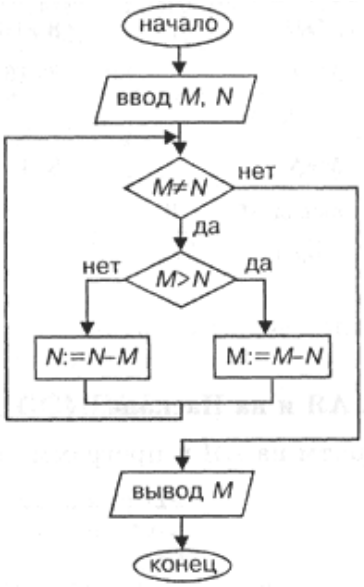


Рисунок 1.5 – Решение алгоритма Евклида блок-схемой

1.6 Алгоритм нахождения наименьшего элемента числовой последовательности

Пример 1.5 Опишем алгоритм нахождения наименьшего элемента последовательности с использованием диаграмм Насси-Шнейдермана.

Решение Описание алгоритма с использованием диаграмм Насси-Шнейдермана:

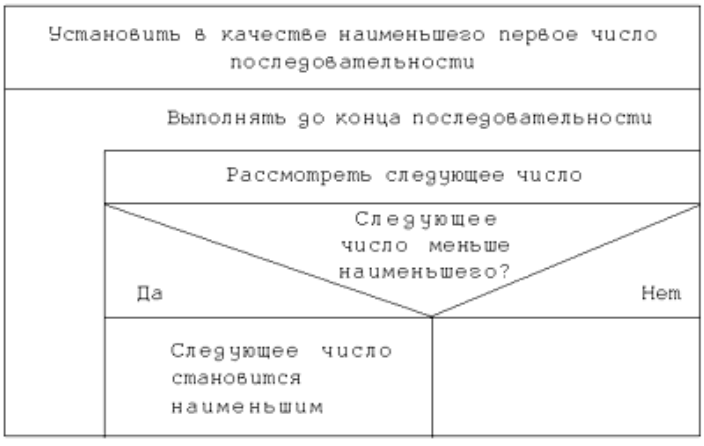


Рисунок 1.6 - Описание алгоритма с использованием диаграмм Насси-Шнейдермана

1.7 Алгоритм сортировки числовой последовательности

Пример 1.6 Сортировка методом пузырька, который также называют методом простого обмена. Рассмотрим алгоритм и особенности этой сортировки.

Пройдем по массиву, сравнивая элементы попарно, т.е. первый элемент со

вторым, второй с третьим, и т.д. до последнего. При проходе будем проверять упорядоченность и, если она нарушена, т.е. если значение некоторого элемента больше значения следующего элемента, то поменяем эти значения местами. Выполняя новые проходы, каждый раз сокращая количество просматриваемых элементов на 1, в конечном итоге отсортируем все значения.

Пример 1.7 Сортировка Хоара (быстрая сортировка).

Основная идея алгоритма состоит в том, что случайным образом выбирается некоторый элемент массива *x*, после чего массив просматривается слева, пока не встретится элемент *a[i]* такой, что *a[i]* > *x*, а затем массив просматривается справа, пока не встретится элемент *a[j]* такой, что *a[j]* < *x*. Эти два элемента меняются местами, и процесс просмотра, сравнения и обмена продолжается, пока мы не дойдем до элемента *x*. В результате массив окажется разбитым на две части – левую, в которой значения ключей будут меньше *x*, и правую со значениями ключей, большими *x*. Далее процесс рекурсивно продолжается для левой и правой частей массива до тех пор, пока каждая часть не будет содержать один элемент.

Задачи и упражнения для выполнения на занятии

Общее задание. В среде Microsoft Office Visio 2007 составить блок-схемы разработанных алгоритмов. Оценить сложности разработанного алгоритма.

1 Составить алгоритм обмена значениями двух переменных a и b:

a) с использованием вспомогательной переменной;

b) с использованием арифметических операций и без использования вспомогательной переменной;

c) с использованием операции исключающего ИЛИ и без использования вспомогательной переменной.

2 Составить алгоритм нахождения минимального и максимального значения переменных *a* и *b*.

3 Составить алгоритм нахождения минимального элемента в однородной последовательности.

4 Составить алгоритм нахождения позиции заданного элемента в однородной неупорядоченной (упорядоченной) последовательности.

5 Составьте блок-схему алгоритмов сортировки элементов однородной последовательности указанным методом:

a) пузырьковая сортировка (сортировка пузырьком (англ. Bubble sort)). Сложность алгоритма: $O(n^2)$. Для каждой пары индексов производится обмен, если элементы расположены не по порядку.

b) быстрая (Quicksort или сортировка Чарльза Хоара) сортировка в варианте с минимальными затратами памяти. Сложность алгоритма: O(n log n) – среднее время, $O(n^2)$ – худший случай. Разбиение исходного набора данных на две половины так, что любой элемент первой половины упорядочен относительно любого элемента второй половины; затем алгоритм применяется рекурсивно к каждой половине.

Задачи и упражнения для выполнения дома

1 Составить алгоритм нахождения максимального элемента в однородной последовательности.

2 Составить алгоритм нахождения позиции заданной подстроки в строке:

a) алгоритм Кнута-Морриса-Пратта поиска подстроки в строке;
b) алгоритм Рабина-Карпа поиска подстроки в строке;
c) алгоритм Бойера-Мура поиска подстроки в строке.

3 Составить алгоритм нахождения фальшивой монеты из N монет. Известно, что фальшивая монета одна и по весу отличается от остальных монет, причем неизвестно, легче она или тяжелее.

Индивидуальная работа Время выполнения – 2 часа.

Задания к лабораторной работе

1 Опишите алгоритм сортировки элементов однородной последовательности указанным (в зависимости от варианта) методом

Вариант 1	Сортировка выбором (Selection sort): поиск наименьшего или наибольшего элемента и помещения его в начало или конец упорядоченного списка.	Вариант 12	сортировка слиянием (Merge sort): выстраиваем первую и вторую половину списка отдельно, затем сливаем упорядоченные списки.
Вариант 2	Сортировка Шелла (Shell sort): попытка улучшить сортировку вставками.	Вариант 13	Поразрядная сортировка (вариант most significant digit (MSD)).
Вариант 3	Сортировка бинарными слияниями.	Вариант 14	Плавная сортировка (Smoothsort).
Вариант 4	Сортировка вставками (Insertion sort): определяем где текущий элемент должен находиться в упорядоченном списке и вставляем его на это место.	Вариант 15	Stooge sort (Сортировка по частям, блуждающая сортировка): рекурсивный алгоритм сортировки с временной сложностью.
Вариант 5	Топологическая сортировка.	Вариант 16	Сортировка расчёской (Comb sort).
Вариант 6	Пирамидальная сортировка (Сортировка кучи, Heapsort): превращаем список в кучу, берём наибольший элемент и добавляем его в конец списка.	Вариант 17	Introsort (интроспективная сортировка): сочетание быстрой и пирамидальной сортировки, которая применяется в случае, если глубина рекурсии превышает log(n).
Вариант 7	Шейкерная (Cocktail sort, bidirectional bubble sort) сортировка (сортировка перемешиванием).	Вариант 18	Сортировка с помощью бинарного дерева (англ. Tree sort).
Вариант 8	Patience sorting: находит самую длинную увеличивающуюся подпоследовательность.	Вариант 19	Блочная сортировка (корзинная сортировка, Bucket sort).
Вариант 9	Быстрая с составными ключами	Вариант 20.	Топологическая сортировка
Вариант 10	Поразрядная сортировка (вариант least significant digit (LSD)).	Вариант 21	Гномья сортировка: имеет общее с сортировкой пузырьком и сортировкой вставками.
Вариант 11	Сортировка подсчётом (Counting sort).	Вариант 22	Сортировка простым выбором

Рекомендуемая литература: Кнут Д. Искусство программирования [Текст] / Д. Кнут : в 4 т. Т.1, 3.

Лабораторная работа №2 **РЕКУРСИВНЫЕ ФУНКЦИИ**

Краткие сведения

Пусть имеется класс функций типа $y\ (x_1, x_2, \ldots, x_n)$, особенностями которых является то, что все аргументы функции $x_1, \ldots, x_n$ целочисленные, и значение функции y также выражается целым числом. Другими словами, рассматриваются функции, аргументы и значения которых дискретны.

Функция $y(x_1, x_2, \ldots, x_n)$ называется *эффективно вычислимой*, если существует алгоритм, позволяющий вычислить ее значение по известным значениям аргументов.

Будем строить различные классы функций следующим образом:

- сначала выберем набор простых базовых функций;
- зафиксируем способы построения одних функций из других (функционалы);
- класс функций определяется набором базовых функций и методов построения всех остальных функций.

Выберем в качестве базовых указанный набор функций:

$\boldsymbol{S^1(x) = x + 1}$ – это одноместная (т.е. имеет один аргумент) функция *непосредственного следования* или *взятие следующего за данным числа*;

$0^n(x_1, x2, \ldots, x_n) = 0$ – n-местная функция, *тождественного равенства 0*;

$\boldsymbol{I_m^n(x_1, x_2, \ldots, x_n) = x_m}$ – n-местная функция *тождественного повторения* значения одного из своих аргументов или *проекция* (где $0 < m \le n$), например, $\boldsymbol{I_2^n(x_1, x_2, \ldots, x_n) = x_2}$.

Зададим функционалы – *суперпозицию* и *примитивную рекурсию*.

Суперпозиция. Пусть имеются n функций m аргументов каждая: $\boldsymbol{f_1(x_1,\ldots,x_m), f_2(x_1,\ldots,x_m),\ldots, f_n(x_1,\ldots,x_m),}$ Пусть имеются функция $\boldsymbol{g(x_1,\ldots,x_n)}$ с n аргументами, тогда обозначенная как $\boldsymbol{Sup^{n+1}(g, f_1, f_2,\ldots, f_n)}$, есть новая функция h с n аргументами, определенная следующим равенством: $\boldsymbol{h^n(x_1,\ldots,x_n) = g^n(f_1(x_1,\ldots,x_m), f_2(x_1,\ldots,x_m),\ldots, f_n(x_1,\ldots,x_m))}$. Говорят, что функция h^n получена из функций g^n, f_1, …, f_n ***суперпозицией*** (или подстановкой). Символически подстановка обозначается как: $\boldsymbol{Sup^{n+1}(g, f_1,\ldots, f_n)}$, где индекс сверху обозначает количество функций, подставляемых в качестве аргументов.

Пример 2.1 Операция подстановки (суперпозиции). Пусть даны функция $f(x) = x + 1$ и функция $g(y) = y + 1$, тогда подставив вместо переменной x функцию $g(y)$, получим новую функцию: $f(y) = y + 2$.

Пусть заданы какие-либо числовые частичные функции: n-местная $g(x_1, \ldots, x_n)$ и $(n+2)$-местная $h(x_1, \ldots, x_n, k, y)$. Говорят, что $(n+1)$-местная частичная функция f образуется из функций g и h посредством ***примитивной рекурсии***, если для всех натуральных значений $x_1, \ldots, x_n, y$ справедливо:

$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$, начальное условие.

$f(x_1, \ldots, x_n, y + 1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, 0))$. рекурсивный шаг…….. *(1)*

Пример 2.2 Операция примитивной рекурсии. Пусть $\boldsymbol{h(x)}$ – известная вычислимая функция, т.е. известен алгоритм вычисления этой функции для произвольного x. Определим новую функцию так:

$$f(x) = \begin{cases} f(0) = a, \\ f(x+1) = f(x, h(x)). \end{cases}$$

Получаем определение функции $f(x)$ с помощью операции примитивной рекурсии. Так, например, если

$$\begin{cases} f(x,1) = x \\ f(x, y+1) = f(x, y) + x \end{cases},$$

то получаем определение целочисленного умножения, т.к. $f(3,3) = f(3,2)+3 = f(3,1)+3+3=3+3+3=9$.

Примитивно-рекурсивной функцией называется функция, которая является одной из базовых функций или может быть построена из базовых функций S^1, 0^n и I_m^n с помощью конечного числа применений функционалов суперпозиции и примитивной рекурсии.

Пример 2.3 Доказать, что двуместная функция $f(x, y) = x + y$ является примитивно-рекурсивной.

Решение. Данная функция может быть представлена в форме *(1)*:

$f(x_1, \ldots, x_n, 0) = g(x_1, \ldots, x_n)$,

$f(x_1, \ldots, x_n, y+1) = h(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, 0))$.

Т.е.

$f(x, \boldsymbol{0}) = x + \boldsymbol{0} = x =$

$f(x, \boldsymbol{y+1}) = x + \boldsymbol{(y+1)} = (x+y) +1 = S^1(x + y)$.

Следовательно, функция $f(x, y)$ образуется из примитивно-рекурсивных функций операцией примитивной рекурсии, и, следовательно, она примитивно рекурсивна.

Пример 2.4 Показать примитивную рекурсивность функции $max(x,y)$.

Решение

$$\begin{cases} \max(0, a) = a \\ \max(b,0) = b \\ \max(a+1, b+1) = \max(a, b) + 1 \end{cases}$$

Более сложной оказывается задача отыскания наименьшего из тех значений y, при которых $f(x, y) = 0$. Поскольку результат решения такой задачи зависит от x, то и наименьшее y является функцией x. Примем обозначение $\varphi(x) = \mu_y[f(x, y) = 0]$ (читается: «наименьшее y такое, что $f(x,y)=0$», а μ_y называет μ-оператором или ***оператором минимизации***). Подобным же образом определяется функция многих переменных $\varphi(x_1, \ldots, x_n) = \mu_y[f(x_1, \ldots, x_n, y) = 0]$.

Пример 2.5 Найти результат операции минимизации к функции $g(x, y) = |x - 2 \cdot y|$.

Решение Результатом является функция $f(x)$ вида:

$f(x) = \mu_y[g(x, y) = 0] = \mu_y[|x-2 \cdot y| = 0]$.

1) При $x = 0$: $g(\mathbf{0}, 0) = | 0 - 2 \cdot \mathbf{0} | = 0$, т.е. $f(0) = \mathbf{0}$

2) При $x = 1$: $g(\mathbf{1}, 0) = | 1 - 2 \cdot 0 | = 1 \neq 0$,

$g(\mathbf{1}, 1) = | 1 - 2 \cdot 1 | = 1 \neq 0$,

$g(\mathbf{1}, 2) = |\ 1 - 2 \cdot 2\ | = 3 \neq 0,$

$g(\mathbf{1}, 3) = |\ 1 - 2 \cdot 3\ | = 5 \neq 0, \ldots$ т.е. $f(1) =$ *не определено*

3) При $x = 2$: $g(\mathbf{2}, \mathbf{0}) = |\ 2 - 2 \cdot \mathbf{0}\ | = 2 \neq 0,$

$g(\mathbf{2}, 1) = |\ 2 - 2 \cdot \mathbf{1}\ | = 0,$ т.е. $f(2) = \mathbf{1}$

4) При $x = 3$: $f(3) =$ *не определено*

5) При $x = 4$: $f(4) = \mathbf{2}$

Ответ: $f(x) = \mu_y[|x - 2 \cdot y| = 0] = [x/2]$

Пример 2.6 Пусть $\boldsymbol{f(x, y) = x^y - 2 \cdot x - 3}$. Применим оператор минимизации и получим $\mu_y f(3, y) = 2$.

Частичные функции (функции f от одного или нескольких аргументов, заданные на множестве всех неотрицательных целых чисел или на некоторых его подмножествах), которые можно получить при помощи операций подстановки примитивной рекурсии и минимизации из простейших функций называются ***частично-рекурсивными***.

Гипотеза Черча состоит в том, что класс построенных таким образом частично рекурсивных функций совпадает с классом функций, допускающим алгоритмическое вычисление.

Пример 2.7 Найдем значение $\boldsymbol{x^y}$ так

$$\begin{cases} f(x,1) = x, \\ f(x, y+1) = h(x, f(x, y)) \end{cases} \quad \text{и} \quad \begin{cases} h(1, x) = x, \\ h(y+1, x) = h(y, x) + x. \end{cases}$$

Рассмотрим следующий пример вычисления 3^2.

$f(3,1+1)=h(3,f(3,1))=h(3,3)=h(2+1,3)=h(2,3)+3=h(1+1,3)+3=h(1,3)+3+3=3+3+3=9$

Задачи и упражнения для выполнения на занятии

1 Найти значение: *a)* $Sup^2(S^1, 0^1)$. *b)* $\boldsymbol{Sup^3(I_1^2, I_1^1, 0^1)}$.

2 Представьте функции $f(x, y)=|x-y|$, $g(x, y)=\min(x, y)$ $d(x, y)=\max(x, y)$ в виде суперпозиции сложения и функции усеченной разности вида

$$\boldsymbol{x \dot{-} y = \begin{cases} 0, \text{ если } x \leqslant y, \\ x - y, \text{ если } x > y. \end{cases}}$$

3 Докажите, что следующие функции являются примитивно рекурсивными:

a) $f(x) = x + 4$;

b) $f(x) = |3 \cdot x - 5|$;

c) $f(x) = 6 \cdot x^2 + 2 \cdot x - 8$;

d) $f(x) = |6 \cdot x^2 + 2 \cdot x - 8|$;

e) $f(x, y) = x+y$;

f) $f(x, y) = x \cdot y$;

g) $f(x) = 2^x$;

h) $f(x) = x!$

i) *Знак числа:* $\boldsymbol{sg(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \end{cases}}$

j) *Дополнение:* $\boldsymbol{\overline{sg}(x) = \begin{cases} 1, & x = 0 \\ 0, & x > 0 \end{cases}}$

k) $\boldsymbol{f(x, y) = rect(x, y) = \begin{cases} x \bmod y, & y \neq 0, \\ x, & y = 0. \end{cases}}$

4 Напишите примитивно-рекурсивное описание функций:

a) $f(x) = x + 7$;

b) $f(x) = 5 \cdot x + 19$;

c) $f(x)$, удовлетворяющей условию $\begin{cases} f(x, 0) = x \\ f(x, y+1) = f(x, y) - 1 \end{cases}$

d) $f(x, y) = 5 \cdot x + 3 \cdot y$;

e) $f(x, y) = 5 \cdot sg(x) + x \cdot y$, где

$$sg(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \end{cases}$$

5 Какая функция получается из функции $g(x)$ и $h(x, y, z)$ с помощью операции примитивной рекурсии:

a) $g(x) = x$; $h(x, y, z) = z^x$.

b) $g(x) = 1$; $h(x, y, z) = z \cdot (x+1)$.

c) $g(x) = x - y$; $h(x, y, z, t) = 2 \cdot (x + y) \cdot f(x, y, z)$?

6 Найти результат применения операции минимизации к функции:

a) $g(x, y) = | x - 2 \cdot y |$.

b) $f(x,y)=x-y+1$;

c) $f(x,y)=y-x$;

d) $f(x,y)=x+y+1$;

e) $f(x,y)=8$;

f) $f(x,y)=x \times y$;

g)
$$f(x, y) = \begin{cases} x - y, & x \geq y, \\ \textit{не определено}, & x < y. \end{cases}$$

7 Докажите, что следующие функции частично-рекурсивны:

$$f(x, y) = \begin{cases} x - y, & x \geq y, \\ \textit{не определено}, & x < y. \end{cases} \qquad f(x, y) = \begin{cases} x / y, & [x / y] \in Z, \\ \textit{не определено}, & \textit{иначе}. \end{cases}$$

$$f(x, y) = \begin{cases} z, & z^y = x, \\ \textit{не определена в остальных случаях}. \end{cases}$$

8 Являются ли частично-рекурсивными или общерекурсивными следующие функции:

a) O(x), S(x), $I_m^n(x_1, x_2, ..., x_n) = x_m$;

b) $g(x, y) = | x - 2 \cdot y |$.

c) $f(x) = x+n$

d) $g(x, y) = x+y$

e) $g(x, y) = x \cdot y$

f) $g(x, y) = x-y$

g) $g(x, y) = x/y$

9 Используя рекурсивные функции постройте:

a) трехместную функцию сложения;

b) двухместную функцию умножения;

c) трехместную функцию умножения;

Задачи и упражнения для выполнения дома

1 Используя рекурсивные функции постройте:

a) n-местную функцию сложения;

b) n-местную функцию умножения.

2 Постройте частичную двухместную функцию деления.

3 Докажите, что если функция $f^n(x_1,...,x_n)$ частично-рекурсивна, то следующие функции частично-рекурсивны:

а) $f_1(x_1,x_2,...,x_n)=f(x_2,x_1,...,x_n)$ (перестановка аргументов);

б) $f_2(x_1,x_2,...,x_n)=f(x_2,,...x_n,x_1)$ (циклическая перестановка аргументов);

в) $f_3(x_1,...,x_n,x_{n+1})=f(x_1,...,x_n)$ (введение фиктивного аргумента);

г) $f_4(x_1,...,x_{n-1})=f(x_1,x_1,...,x_{n-1})$ (отождествление аргументов).

Индивидуальная работа Время выполнения – 2 часа.

Задания к лабораторной работе

1 Докажите, что следующие функции являются примитивно рекурсивными.

2 Какая функция получается из функции *g(x)* и *h(x, y, z)* с помощью операции примитивной рекурсии (для вариантов 1-4)? Является ли следующая функция частично рекурсивной или общерекурсивной (для вариантов 5-24)?

3 Найти результат применения операции минимизации к функции.

Вариант 1

1 $f(x) = x + n$.

2 $g(x) = x$; $h(x, y, z) = x^z$.

3 $f(x, y) = |x - 3 \cdot y|$.

Вариант 2

1 $f(x) = 2 \cdot x + 3 \cdot y$;

2 $g(x) = x$; $h(x, y, z) = x \cdot y + f(x, 0)$.

3 $f(x, y) = x - y + 2$.

Вариант 3

1 $f(x, y) = k \cdot x + n \cdot y$

2 $g(x) = x - y$;
$h(x, y, z, t) = 3 \cdot (x + y) \cdot f(x, y, z)$.

3 $f(x, y) = 3 \cdot x \cdot y$.

Вариант 4

1 $f(x) = x^x$;

2 $g(x,y) = x - y$;
$h(x, y, z, t) = (x + y) \cdot f(x, y, z)$.

3 $f(x, y) = |x - y|$.

Вариант 5

1 $f(x, y) = (x+y)^{n \cdot x \cdot y}$, где $k,n \in N$;

2 $g(x,y) = |x - 3 \cdot y|$.

3 $f(x, y) = x + y + 2$.

Вариант 6

1 $f(x) = 3^{x^2+1}$;

2 $g(x) = 2 \cdot x + n$, *n-const.*

3 $f(x, y) = 2 \cdot x + y + 1$.

Вариант 7

1 $f(x) = x^2 + 3 \cdot x + 2$.

2 $g(x,y) = 2 \cdot x + 3 \cdot y$.

3 $f(x, y) = f(x, y) = x + 2 \cdot y + 1$.

Вариант 8

1 $f(x, y) = 2 \cdot x - y$.

2 $g(x,y) = n \cdot x \cdot y$, *n – const.*

3 $f(x, y) = | y - x |$.

Вариант 9

1 $f(x) = 2 \cdot x + 1$.

2 $g(x,y) = 3 \cdot x - 2 \cdot y$.

3 $f(x, y) = | x^2 - y^2 |$.

Вариант 10

1 $f(x, y) = k \cdot x + 3 \cdot y$.

2 $g(x,y) = (5 \cdot x) / y$.

3 $f(x, y)\ 3 \cdot x - x \cdot y + 4$.

Вариант 11

1 $f(x, y) = x^y$.

2 $$f(x, y) = \begin{cases} x - y, & x \geq y, \\ \text{не определено}, & x < y. \end{cases}$$

3 $f(x, y,z) = x \cdot y + z + 5$.

Вариант 12

1 $f(x, y) = (x + y)^{x \cdot y}$.

2 $$f(x, y) = \begin{cases} x / y, & [x / y] \in Z, \\ \text{не определено}, & \text{иначе}. \end{cases}$$

3 $f(x, y,z) = x \cdot y + z + 8$.

Вариант 13

1 $$f(x, y) = min(x, y) = \begin{cases} x, & x < y, \\ y, & x \geq y. \end{cases}$$

2 $g(x,y) = [x / y]$.

3 $f(x, y) = x \cdot y - 2$.

Вариант 14

1 $$f(x, y) = max(x, y) = \begin{cases} x, & x \geq y, \\ y, & x < y. \end{cases}$$

2 $g(x,z) = |2 \cdot z^2 - x^2|$.

3 $f(x, y) = x/y + 5$.

Вариант 15

1 $y = \lfloor \sqrt{x} \rfloor$.

2 $g(x,y) = |y^2 - 3 \cdot x^2|$.

3 $f(x, y, z) = x \cdot z + y + 5$.

Вариант 16

1. $f(x, y) = 2^x \cdot (2 \cdot y + 1) - 1$;

2. $g(x,y,z) = x \cdot y + z + 1$.

3. $f(x, y) = 3 \cdot y^2 + x \cdot y + 4$.

Вариант 17

1. $f(x, y) = |x - y| = \begin{cases} x - y, & x \geq y, \\ y - x, & x < y. \end{cases}$
2. $g(x,y) = |x^2 - y^2|$
3. $f(x, y) = |3 \cdot x - 2 \cdot y|$

Вариант 18

1 $f(x) = x!$, *где*

$$x! = \begin{cases} 1, & x = 0, \\ 1 \cdot 2 \cdot \ldots \cdot x, & x > 0. \end{cases}$$

2 $g(x)$, *x-ое простое число*

3 $f(x, y) = |2 \cdot x - y|$

Вариант 19

1 $f(x) = 5 \cdot x! + 8$, *где*

$$x! = \begin{cases} 1, & x = 0, \\ 1 \cdot 2 \cdot \ldots \cdot x, & x > 0. \end{cases}$$

2 $f(x, y) = \begin{cases} z, & z^y = x, \\ \text{не определена в остальных случаях.} \end{cases}$

3 $f(x, z) = |z^2 - x^2|$

Вариант 20

1 *Ограниченное вычитание:* $x \dot{-} y = \begin{cases} x - y, & x > y \\ 0, & x \leq y \end{cases}$

2 $g(x,z) = |z^2 - x^2|$

3 $f(x, y) = |y^2 - x^2|$

Вариант 21

1 $\tau(x) = \begin{cases} \textbf{число делителей числа}, \\ 0, & x = 0. \end{cases}$

2 $g(x,y) = 3 \cdot x^2 + x \cdot y + 4$

3 $f(x,y)$,*удовлетворяющей условиям:* $\begin{cases} f(x,0) = x, \\ f(x, y+1) = x \cdot y + f(x, y). \end{cases}$.

Вариант 22

1 $\sigma(x) = \begin{cases} \textbf{сумма делителей числа}, \\ 0, & x = 0. \end{cases}$

2 $q(x,z) = 2 \cdot z^2 + x \cdot z - 8.$

3 $f(x,y)$,*удовлетворяющей условиям:* $\begin{cases} f(x,0) = 1, \\ f(x, y+1) = y \cdot f(x, y). \end{cases}$.

Лабораторная работа №3-4. АБСТРАКТНАЯ МАШИНА ТЬЮРИНГА

Краткие сведения

Описание машины Тьюринга (МТ)

Машина Тьюринга представляет собой автомат, имеющий бесконечную в обе стороны ленту, считывающую головку и управляющее устройство (УУ). Управляющее устройство может перемещаться влево и вправо по ленте, читать и записывать в ячейки ленты символы некоторого конечного алфавита. Выделяется особый пустой символ, заполняющий все клетки ленты, кроме тех из них (конечного числа), на которых записаны входные данные. УУ работает согласно правилам перехода, которые представляют алгоритм, реализуемый данной МТ. Каждое правило перехода предписывает МТ, в зависимости от

текущего состояния и наблюдаемого в текущей клетке символа, записать в эту клетку новый символ, перейти в новое состояние и переместиться на одну клетку влево или вправо. Некоторые состояния МТ могут быть помечены как терминальные, и переход в любое из них означает конец работы (остановку алгоритма). Таким образом, за 1 такт работы МТ может считать символ, записать вместо него новый или оставить его без изменения и сдвинуть головку на одну ячейку влево или вправо или оставить ее на месте.

Конкретная машина Тьюринга задается перечислением:

– элементов множества букв алфавита A = {a_0, a_1, …, a_m}, где a_0 – символ пустой ячейки;

– элементов множества состояний Q = {q_0, q_1, ..., q_n}, где q_0 – состояние остановки: попав в него, машина прекращает работу; q_1 – начальное состояние: в этом состоянии машина начинает работать;

– набором правил, по которым работает машина. Правила имеют вид: $\mathbf{q_i a_t \rightarrow q_j a_p d_k}$ (если головка находится в состоянии q_i и в обозреваемой ячейке записана буква a_t, то головка переходит в состояние q_j, в ячейку вместо a_t записывается a_p, головка делает движение d_k, которое имеет три варианта: на ячейку влево (L), на ячейку вправо (R), остаться на месте (H)).

Описание машины Тьюринга совокупностью команд, таблицей соответствия и в виде графа

Пример 3.1 Рассмотрим совокупность команд МТ, которая инвертирует входную цепочку, записанную с использованием нулей и единиц. Пусть алфавит МТ задан множеством A={0, 1, ε}, где ε соответствует пустой ячейке. Пусть число состояний УУ задано в виде множества Q = {q_0, q_1, q_z}. Если, например, начальная конфигурация имеет вид $q_0\underline{1}10011$, то конечная конфигурация после завершения операции инвертирования имеет вид $q_z\underline{0}01100$.

Идея решения В стандартной начальной конфигурации головка стоит над первым символом слева, и УУ находится в начальном состоянии. На следующем такте МТ, не меняя своего состояния, заменяет символ 0 на 1 или 1 на 0, сдвигается вправо на один символ. После просмотра всей цепочки под головкой оказывается символ, указывающий на пустую ячейку. В этом случае МТ переходит в новое состояние и сдвигается влево на один символ. На последующих тактах УУ не меняет своего состояния, оставляет без изменения символ под головкой и перемещается влево до тех пор, пока не встретит пустую ячейку. Встретив пустую ячейку, МТ переходит в заключительное состояние и перемещается вправо на один символ, переходя в стандартную заключительную конфигурацию q_z.

Для решения задачи МТ будет порождена следующей последовательностью команд:

$q_0 1 \rightarrow q_0 0 R$,
$q_0 0 \rightarrow q_0 1 R$,
$q_1 0 \rightarrow q_1 0 L$,
$q_1 1 \rightarrow q_1 1 L$,
$q_0 \varepsilon \rightarrow q_1 \varepsilon L$,

$q_1\varepsilon \rightarrow q_z\varepsilon R$.

МТ для рассмотренного примера инвертирования цепочки, состоящей из символов 0 и 1, будет представлена в виде графа следующим образом (рисунок 3.1):

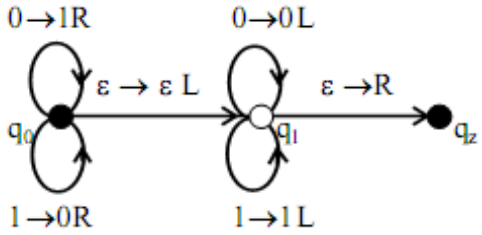


Рисунок 3.1 – МТ в виде графа

МТ из рассмотренного примера инвертирования цепочки, состоящей из символов 0 и 1, будет представлена таблицей соответствия МТ (таблица 3.1).

Таблица 3.1 – Таблица соответствия МТ

Состояние	Символы		
	0	**1**	**ε**
q_0	$q_0$1R	$q_0$0R	$q_1\varepsilon$L
q_1	$q_1$0L	$q_1$1L	$q_z\varepsilon$R

Задачи и упражнения для выполнения на занятии

1 Построить таблицу соответствия и функциональную схему для машины Тьюринга, которая удаляет из числа все нули, например, число 1001110 преобразует к виду 1111.

2 Построить таблицу соответствия и функциональную схему для машины Тьюринга, которая меняет местами соседние два элемента попарно. Пример. Исходное число 011001 заменяется на 100110.

3 Постройте функциональные схемы для машины Тьюринга, вычисляющие простейшие арифметические операции (сложение, вычитание, умножение, деление) двух чисел.

4 Постройте программу для машины Тьюринга вычисляющую следующую функцию $f(x) = x + 3$.

Указание: Взять множество состояний Q = {q_0, q_1, q_2}. Число x на ленте МТ записывается в десятичной системе счисления. Состояние q_1 заменяет последнюю цифру числа x, если эта цифра меньше 7, цифрой, на 3 единицы большей, и переходит в стоп-состояние. Если последняя цифра числа x равна 7, то ее заменить на 0 и перейти влево в состояние q_2. Состояние q_2 добавляет к следующему разряду 1. Если же последняя цифра числа x равна 8, то ее заменить на 1 и перейти влево в состояние q_2. Если же последняя цифра числа x равна 9, то ее заменить на 2 и перейти влево в состояние q_2.

5 Вычисляет ли машина Тьюринга функцию $sign(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \end{cases}$ в алфавите {Λ, 1} с программой в виде таблицы соответствия (таблица 3.2):

Таблица 3.2 – Таблица соответствия для задачи 5

Символы	Состояние			
	q_1	**q_2**	**q_3**	**q_4**
Λ	q_2ΛL	q_0ΛR	q_4ΛR	q_4ΛR
1		$q_3$1L	q_3ΛL	$q_0$1H

6 По заданной совокупности команд машины Тьюринга Т и начальной конфигурации К найти заключительную конфигурацию.

$q_0 1 \to q_0 1R$,	$q_1 0 \to q_z 1E$,	$q_0 0 \to q_0 1L$,	$q_1 1 \to q_0 0R$,
$q_0 0 \to q_1 1R$,	$q_1 1 \to q_1 1L$,	$q_0 1 \to q_1 1R$,	$q_1 0 \to q_z 0L$,
а) $K = 1101 q_0 01$;	б) $K = 101 q_0 010$.	а) $K = 1 q_1 0111$;	б) $K = 1 q_1 1111$.

7 Машины Тьюринга T_1 и T_2 заданы таблицами соответствия (таблицы 3.3 и 3.4). Построить композицию $T_1 \cdot T_2$ машин Тьюринга T_1 и T_2 по паре состояний (q_{1z}, q_{20}) и найти результат применения $T_1 \cdot T_2$ к слову D=11000101001, D=10100111110.

Таблица 3.3 – Таблица соответствия МТ T_1

T_1:

	q_{10}	q_{11}	q_{12}
0	$q_{11}0R$	$q_{12}0R$	$q_{1z}1L$
1	$q_{10}1R$	$q_{10}1R$	

T_2:

	q_{20}	q_{21}	q_{22}
0	$q_{21}0L$	$q_{22}0L$	$q_{2z}0R$
1	$q_{20}1L$	$q_{21}1L$	$q_{22}1L$

8 Машина Тьюринга Т задана графом:

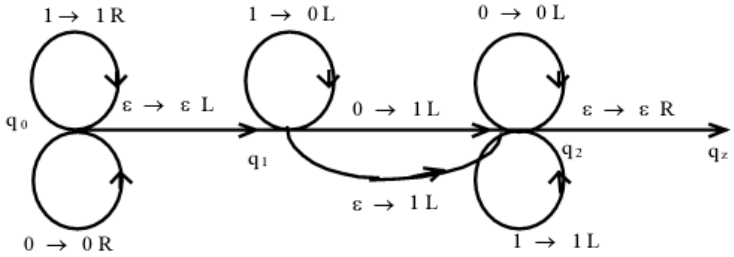


Рисунок 3.2 – Граф МТ

Постройте таблицу соответствия.

Задачи и упражнения для выполнения дома

1 Построить таблицу соответствия, функциональную схему машины Тьюринга, которая получает обратный порядок записи числа, например, исходное число 111001, результат 100111.

2 Построить таблицу соответствия и функциональную схему машины Тьюринга, распознающую слова, заканчивающиеся тремя подряд идущими единицами.

3 Построить граф, таблицу соответствия и функциональную схему машины Тьюринга, которая получает обратный порядок записи числа, например, исходное число 111001, результат 100111.

Индивидуальная работа Время выполнения – 4 часа.

Задания к индивидуальной работе

1 Разработайте функциональную схему машины Тьюринга для решения указанной задачи.

2 Постройте программу (в виде протокола, таблицы соответствия и в виде графа) машины Тьюринга, вычисляющую следующую функцию.

3 Используя базис элементарных машин с помощью операций композиции, ветвления и зацикливания постройте машины, вычисляющие соответствующие функции:

Замечание. Для вариантов 1-10 дополнительно следует использовать и машины M_1, M_2, M_3, вычисляющие соответственно функции $f(x)=2\cdot x$, $f(x, y)=x+y$ и $f(x,y)= x \dot{-} y$,

Вариант 1

1 Дано число n в восьмеричной системе счисления. Разработать программу для машины Тьюринга, которая увеличивала бы заданное число n на 1.

2 $f(x) = x + 5$,

3 $f(x, y)= x \cdot y$.

Вариант 2

1 На информационной ленте машины Тьюринга содержится массив символов «+». Необходимо разработать функциональную схему машины Тьюринга, которая каждый второй символ «+» заменит на «–». Каретка в состоянии q_0 находится где-то над указанным массивом.

2 $f(x) = 4\cdot x$.

3 $f(x, y)=2\cdot x + y$

Вариант 3

1 Дан массив из открывающихся и закрывающихся скобок. Построить программу для машины Тьюринга, которая удаляла бы пары взаимных скобок. Например, дано : «) (( ) (( ) », надо получить : «) . . . ((. ».

2 $f(x) = 2 – x$.

3 $f(x, y)= 2 \cdot (x \div y)$, где знак «÷» – знак ограниченного вычитания

$$x \div y = \begin{cases} x - y, & x > y \\ 0, & x \le y \end{cases}$$

Вариант 4

1 Дана десятичная запись натурального числа $n>1$. Разработать программу для машины Тьюринга, которая уменьшала бы заданное число n на 1. При этом запись числа n–1 не должна содержать левый 0, например, 100–1=99, а не 099. Начальное положение головки – правое.

2 $f(x) = 5 – x$.

3 $f(x, y)= [x / y]$.

Вариант 5

1 На информационной ленте МТ в трех секциях в произвольном порядке записаны 3 различные буквы: «A», «B» и «C». Каретка в состоянии q_0 обозревает букву, расположенную справа. Необходимо составить функциональную схему МТ, которая поменяет местами крайние буквы.

2 $f(x) = x – 3$.

3 $f(x)= 2\cdot x + 5$.

Вариант 6

1 Дана строка из букв «*a*» и «*b*». Разработать программу для машины Тьюринга, которая переместит все буквы «*a*» в левую, а буквы «*b*» – в правую части строки. Каретка находится над крайним левым символом строки.

2 $f(x) = |x – 3|$.

3 $f(x)= x \div 3$, где знак «÷» – знак ограниченного вычитания

$$x \div y = \begin{cases} x - y, & x > y, \\ 0, & x \le y \end{cases}$$

Вариант 7

1 На ленте машины Тьюринга находится число, записанное в десятичной системе счисления. Умножить это число на 2, если каретка находится над крайней левой цифрой числа.

2 $f(x) = 3 \cdot x - 2$.

3 $f(x,y)$, удовлетворяющей условиям

$$\begin{cases} f(x,0) = 2 \cdot x, \\ f(x,y+1) = x \dot{-} f(x,y). \end{cases}$$ где $$x \dot{-} y = \begin{cases} x - y, & x > y \\ 0, & x \le y \end{cases}$$

Вариант 8

1 Даны два целых положительных числа в десятичной системе счисления. Сконструировать машину Тьюринга, которая будет находить разность этих чисел, если известно, что первое число больше второго, а между ними стоит знак «–». Каретка находится над левой крайней цифрой левого числа.

2 $f(x) = |k \cdot x - 3|$.

3 $f(x, y) = 2 \cdot (x + y) \div (x \div y)$, где знак «$\div$» – знак ограниченного вычитания

$$x \div y = \begin{cases} x - y, & x > y, \\ 0, & x \le y \end{cases}$$

Вариант 9

1 Дана конечная последовательность меток, записанных в клетки ленты подряд, без пропусков. Необходимо разработать программу для машины Тьюринга, которая будет записывать в десятичной системе счисления число этих меток.

2 $f(x) = x - k$.

3

$$f(x,y) = rect(x, y) = \begin{cases} x \bmod y, & y > 0, \\ x, & y = 0. \end{cases}$$

Вариант 10

1 Сконструировать машину Тьюринга, которая выступит в качестве двоично-восьмеричного дешифратора.

2 $f(y) = 2 \cdot y - 1$.

3 $f(x) = \mu y[(x + y) \div 5 = 0]$, где $$x \div y = \begin{cases} x - y, & x > y, \\ 0, & x \le y \end{cases}$$

Вариант 11

1 На ленте машины Тьюринга находится целое положительное число, записанное в десятичной системе счисления. Найти произведение этого числа на число 11. Каретка обозревает крайнюю правую цифру числа.

2 $f(y) = |2 \cdot y - 1|$.

3 $f(x) = x + 5$.

Вариант 12

1 Даны два натуральных числа n и m, заданных в унарной системе счисления. Числа n и m представлены наборами символов «|», разделенных «\». В конце набора стоит «=». Разработать машину Тьюринга, которая будет производить деление нацело двух натуральных чисел n и m и находить остаток от деления. При этом результат должен быть записан следующим образом:

после «=» должен находиться набор символов «|» частного (он может быть и пустым), после чего ставится знак «(», за которым следует набор символов «|» остатка от деления *n* на *m*.

2 $f(x) = |x - k|$.

3 $f(x) = 3 \cdot x + 5$.

Вариант 13

1 Даны два натуральных числа *m* и *n*, представленных в унарной системе счисления. Соответствующие наборы символов «|» разделены «–», вслед за последним символом набора *n* стоит знак «=». Разработать машину Тьюринга, которая будет находить разность чисел m и n. При этом результат должен быть записан следующим образом: если m>n , то справа от «=» должны стоять знак «+» и набор символов «|» в количестве *m–n*; если *m=n*, то справа от знака «=» должна стоять пустая клетка; если *m<n*, то справа от «=» должны стоять знак «–» и набор символов «|» в количестве *n–m*.

2 $f(x) = 2 \cdot x + 3$.

3 $f(x) = x \div 5$, где $x \div y = \begin{cases} x - y, & x > y, \\ 0, & x \le y \end{cases}$

Вариант 14

1 Даны два целых положительных числа в различных системах счисления, одно – в троичной системе, другое – в десятичной. Разработать машину Тьюринга, которая находит сумму этих чисел в десятичной системе счисления.

2 $f(y) = k \cdot y + 1$.

3 $sg(x) = \begin{cases} 0, & x = 0, \\ 1, & x > 0 \end{cases}$

Вариант 15

1 Найти произведение двух натуральных чисел *m* и *n*, заданных в унарной системе счисления. Соответствующие наборы символов «|» разделены знаком «*», справа от последнего символа правого члена стоит знак «=». Поместить результат умножения этих чисел вслед за знаком «=».

2 $f(x) = x^2$,

3 $f(x) = C_5^1(x)$.

Вариант 16

1 На информационной ленте машины Тьюринга находится десятичное число. Найти результат целочисленного деления этого числа на 2.

2 $f(y) = \cdot y + 2$.

3 $\overline{sg}(x) = \begin{cases} 1, & x = 0 \\ 0, & x > 0 \end{cases}$

Вариант 17

1 На информационной ленте машины Тьюринга находится массив, состоящий только из символов *A* и *B*. Сжать массив, удалив из него все элементы *B*.

2 $f(x, y) = x / y$.

3 $f(x, y) = I_5^1(x, y) + 1$

Вариант 18

1 На ленте машины Тьюринга находится десятичное число. Определить делится это число на 5 без остатка. Если делится, то записать справа от числа слово «да», если нет – «нет». Каретка находится где-то над числом.

2 $f(x) = x^2 - 1$.

3 $$f(x,y) = max(x,y) = \begin{cases} x, & x > y, \\ y, & x \le y. \end{cases}$$

Вариант 19

1 Число n задано на ленте машины Тьюринга массивом меток. Преобразовать значение n по формуле: $\varphi(n) = \left[\frac{5n}{4}\right] + 2$. Каретка обозревает крайнюю левую метку.

2 $f(x, y) = x \cdot y$.

3 $f(x) = x!$

Вариант 20

1 На ленте машины Тьюринга записано число в десятичной системе счисления. Каретка находится над крайней правой цифрой. Записать цифры этого числа в обратном порядке.

2 Построить машину Тьюринга, вычисляющую нуль-функцию $0^1(x) = 0$ в алфавите {Λ, 1}.

3 $f(x) = 4 \cdot x + 1$.

Вариант 21

1 На ленте машины Тьюринга находится массив 2*N меток. Уменьшить этот массив в 2 раза.

2 Построить машину Тьюринга, вычисляющую функцию выбора $I_1^3(x_1, x_2, x_3) = x_1$.

3 $f(y) = 3 \cdot y + 2$.

Вариант 22

1 Даны два натуральных числа *n* и *m*, представленные в унарной системе счисления. Между этими числами стоит знак «?». Выяснить отношение *m* и *n*, т.е. знак «?» заменить на один из подходящих знаков «>», «<», «=».

2 Построить МТ, вычисляющую функцию выбора $I_3^3(x_1, x_2, x_3) = x_3$.

3 $$f(x,y) = min(x,y) = \begin{cases} y, & x > y, \\ x, & x \le y. \end{cases}$$

Лабораторная работа №5-6. АБСТРАКТНАЯ МАШИНА ПОСТА

Краткие сведения

Машины Поста – абстрактная вычислительная машина, предложенная Эмилем Леоном Постом (Emil L. Post), которая отличается от машины Тьюринга большей простотой. Обе машины «эквивалентны» и были созданы для уточнения понятия «алгоритм».

Описание машины Э. Поста

Машины Поста состоит из каретки (считывающей и/или записывающей головки) и разбитой на секции бесконечной в обе стороны ленты. Каждая секция ленты может быть либо пустой, либо помеченной меткой. За один шаг каретка может сдвинуться на одну позицию влево или вправо, считать, поставить или уничтожить символ в том месте, где она стоит. Ситуации, в которых головка должна наносить метку там, где она уже имеется, или, наоборот, стирать метку там, где ее нет, являются аварийными (недопустимыми).

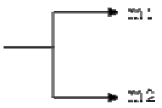
Команда машины Поста имеет следующую структуру: *n* **K** *m*, где *n* – порядковый номер команды, **K** – действие, выполняемое головкой, *m* – порядковый номер следующей команды, подлежащей выполнению.

Программой для машины Поста будем называть непустой список команд, такой что:

1) на n-м месте команда с номером n;

2) номер каждой команды совпадает с номером какой-либо команды списка (таблица 5.1).

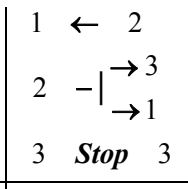
Таблица 5.1 – Команды машины Поста

Пояснение	*команда*
Движение головки на одну клетку вправо	n → m
Движение головки на одну клетку влево	n ← m
Нанесение метки на клетку, над которой находится головка	n **М** m
Стирание метки из клетки, над которой находится головка	n **С** m
Проверка наличия метки в клетке, над которой находится головка. Если метка отсутствует, то управление передается команде m1, а иначе – m2.	n 
Остановка машины	n **Стоп** n

Представление чисел на ленте машины Поста

Число k представляется на ленте машины Поста идущими подряд k+1 метками (одна метка означает число «0»). Между двумя числами делается интервал как минимум из одной пустой секции на ленте.

Пример 5.1 Пусть на ленте имеется запись из нескольких меток подряд и головка находится над самой крайней меткой справа. Требуется перевести головку влево до первой пустой позиции.



Пример 5.2 Пусть требуется прибавить 1 к натуральному числу N. Головка находится где-то слева от меток числа.

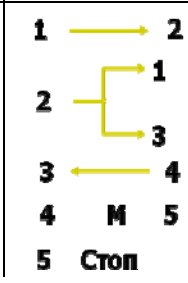


Рисунок 5.1 – Машина Поста для примера 5.2

Задачи и упражнения для выполнения на занятии

1 Напишите программы для машины Поста, вычисляющие результат простейших арифметических операций (сложение, умножение) над двумя числами, разделенными на ленте а) одним пробелом; б) произвольным числом пробелов.

2 Напишите программу для машины Поста, вычисляющую результат деления одного числа на другое. Числа на ленте разделены одним пробелом.

Задачи и упражнения для выполнения дома

1 Напишите программы для машины Поста, вычисляющие результат вычитания над двумя числами, разделенными на ленте а) одним пробелом; б) произвольным числом пробелов.

2 Найти остаток от деления целого неотрицательного числа k на 3, если известно, что каретка находится справа от заданного числа.

Индивидуальная работа Время выполнения – 2 часа.

Задания к лабораторной работе

Напишите эффективную программу решения задачи для машины Поста.

Вариант 1

На информационной ленте машины Поста расположено N массивов меток, отделенных друг от друга свободной ячейкой. Каретка находится над крайней левой меткой первого массива. Определить количество массивов.

Вариант 2

На ленте машины Поста расположен массив из 2*N–1 меток. Составить программу для поиска и удаления средней метки массива.

Вариант 3

Составить программу сложения трех целых неотрицательных чисел *x*, *y* и *z*, расположенных на ленте машины Поста. Каретка расположена над одной из меток числа *x*. Число y находится через одну секцию правее числа *x*, а число *z* – правее числа *y* через одну секцию.

Вариант 4

На ленте машины Поста расположен массив из N меток. Необходимо справа от данного массива через одну пустую секцию разместить массив из 2*N меток. При этом исходный массив может быть стерт с ленты. Каретка машины Поста находится над числом.

Вариант 5

Составить программу сложения произвольного количества целых неотрицательных чисел, записанных на ленте машины Поста на расстоянии одной пустой секции друг от друга. Каретка машины Поста находится над крайней левой меткой левого числа.

Вариант 6

Игра Баше. В игре участвуют двое (человек и машина Поста). Суть игры заключается в следующем: имеется 21 предмет. Первым ходит человек. Каждый из играющих может брать 1, 2, 3 или 4 предмета. Проигрывает тот, кто берет последний предмет. Написать программу, по которой всегда будет выигрывать машина Поста.

Вариант 7

На ленте машины Поста расположен массив из N меток. Метки расположены через пустую ячейку. Нужно сжать массив так, чтобы все N меток занимали N расположенных подряд ячеек.

Вариант 8

На ленте машины Поста находятся n массивов меток. Каретка находится где-то над первым массивом. Все массивы разделены тремя пустыми ячейками. Удалить все четные массивы.

Вариант 9

На ленте машины Поста расположен массив из N меток. Составить программу, определения делится ли число на 3. Если да, то после массива через одну пустую секцию поставить метку. Каретка находится где-то над массивом.

Вариант 10

Задача В.А.Успенского. На информационной ленте либо вправо, либо влево от секции, над которой расположена каретка, находится массив меток. Расстояние до массива выражается конечным числом. Необходимо составить программу поиска массива. Каретка должна быть на первой метке массива.

Вариант 11

На ленте машины Поста находятся n массивов меток. Каретка находится над первым массивом. Удалить все нечетные массивы, если все массивы разделены тремя пустыми ячейками.

Вариант 12

На ленте машины Поста расположены два массива. Составить программу стирания большего из этих массивов.

Вариант 13

На ленте машины Поста находятся два массива в M и N меток. Составить программу выяснения, одинаковы ли массивы по длине.

Вариант 14

Дан массив меток. Каретка обозревает первую пустую ячейку перед началом массива. Раздвинуть массив так, чтобы после каждой метки была пустая ячейка.

Вариант 15

Найти остаток от деления целого неотрицательного числа k на m, если известно, что каретка находится справа от заданного числа k.

Вариант 16

На ленте машины Поста находится n массивов меток. Массивы разделены тремя пустыми ячейками. После последнего массива на расстоянии более трех пустых секций находится одна метка. Количество меток в массивах не меньше двух. Произвести следующую обработку массивов: если количество меток в массиве кратно 3, то стереть метки в данном массиве через одну, иначе массив стереть полностью. Каретка находится над крайней левой меткой первого массива.

Вариант 17

На ленте машины Поста расположен массив из 2*N меток. Составить программу, по которой машина Поста раздвинет на расстояние в одну ячейку две половины данного массива.

Вариант 18

На ленту машины Поста нанесены два массива меток на некотором расстоянии друг от друга. Соединить эти два массива в один. Каретка находится над крайней левой меткой левого массива.

Вариант 19

Найти НОД двух чисел, находящихся на ленте машины Поста. Между этими числами – произвольное количество пустых ячеек. Каретка находится над левой меткой левого числа.

Вариант 20

На ленте машины Поста отмечен массив n меток. Найти число 2·n+1 и проверить, делится ли оно на 3. Если да, то после числа через одну пустую ячейку поставить 2 метки, если нет – поставить 3 метки. Каретка находится над крайней левой отмеченной ячейкой.

Вариант 21

На ленте машины Поста находится массив меток. Каретка находится где-то над массивом (но не над крайней меткой). Стереть все метки, кроме крайних, таким образом, чтобы положение каретки при этом не изменилось.

Вариант 22

Составить программу нахождения разности двух неотрицательных целых чисел a и b, находящихся на ленте машины Поста. Каретка находится над левой меткой левого числа. Неизвестно, какое число больше: a или b.

ЛАБОРАТОРНАЯ РАБОТА №7-8. Нормальные алгорифмы Маркова

Краткие сведения

Определение нормального алгоритма Маркова (НАМ)

Нормальным алгоритмом Маркова (НАМ) называется непустой конечный упорядоченный набор формул подстановки:

$$\begin{cases} \alpha_1 \to \beta_1 \\ \alpha_2 \to \beta_2 \\ \ldots \\ \alpha_k \to \beta_k, \quad k \geq 1 \end{cases}$$

В этих формулах могут использоваться два вида стрелок: обычная стрелка ($\to$) и стрелка «с хвостиком» ($\mapsto$). Формула с обычной стрелкой называется обычной формулой, а формула со стрелкой «с хвостиком» – заключительной формулой. Разница между ними объясняется чуть ниже.

Записать алгоритм в виде НАМ – значит предъявить такой набор формул.

Правила выполнения НАМ

Прежде всего, задается некоторое входное слово P. Где именно оно записано – не важно, в НАМ этот вопрос не оговаривается. Работа НАМ сводится к выполнению последовательности шагов. На каждом шаге входящие в НАМ формулы подстановки просматриваются сверху вниз и выбирается первая из формул, применимых к входному слову P, т.е. самая верхняя из тех, левая часть которых входит в P. Далее выполняется подстановка согласно найденной формуле. Получается новое слово P'.

На следующем шаге это слово P' берется за исходное и к нему применяется та же самая процедура, т.е. формулы снова просматриваются сверху вниз начиная с самой верхней и ищется первая формула, применимая к слову P′, после чего выполняется соответствующая подстановка и получается новое слово P'', и т.д.

Следует обратить особое внимание на тот факт, что на каждом шаге формулы в НАМ всегда просматриваются начиная с самой первой.

Необходимые уточнения:

1 Если на очередном шаге была применена обычная формула (α→β), то работа НАМ продолжается.

2 Если же на очередном шаге была применена заключительная формула (α↦β), то после её применения работа НАМ прекращается. То слово, которое получилось в этот момент, и есть выходное слово, т.е. результат применения НАМ к входному слову.

Пример 7.1 Рассмотрим НАМ для вставки и удаления символов. Пусть $A=\{\varnothing, a, b, c, d\}$. В слове P требуется заменить первое вхождение подслова *bb* на *ddd* и удалить все вхождения символа *c*. Например: *abbcabbca* → *adddabba*.

Решение Отметим, что в НАМ, в отличие от машины Тьюринга, легко реализуются вставки и удаления символов. Вставка новых символов в слово – это замена некоторого подслова на подслово с бóльшим числом символов; например, с помощью формулы *bb→ddd* два символа будут заменены на три символа. При этом не надо заботиться о том, чтобы предварительно освободить место для дополнительных символов, в НАМ слово раздвигается автоматически. Удаление же символов – это замена некоторого подслова на подслово с меньшим числом символов; например, удаление символа с реализуется формулой $c \to \varnothing$ (с пустой правой частью). При этом никаких пустых позиций внутри слова не появляется, сжатие слова в НАМ происходит автоматически. С учётом сказанного нашу задачу должен, казалось бы, решать такой НАМ:

$bb \to ddd$ (1)

$c \to \varnothing$ (2)

Однако это не так. Проверим этот НАМ на входном слове abbcabbca (над стрелками указаны номера применённых формул, а в словах слева от стрелок подчёркнуты для наглядности те части, к которым были применены эти формулы):

$$abbcabbca \xrightarrow{1} adddcabbca \xrightarrow{1} adddcadddca \xrightarrow{2} adddabbca \to \ldots$$

Как видно, заменив первое вхождение bb на ddd, этот НАМ не перешёл сразу к удалению символов c, а стал заменять и другие вхождения bb. Почему? Напомним, что на каждом шаге работы НАМ формулы подстановки всегда просматриваются сверху вниз начиная с первой из них. Поэтому, пока применима первая формула, она и будет применяться, блокируя доступ к остальным формулам. Этот означает, что в НАМ важен порядок перечисления формул подстановки.

Учтём это и переставим наши две формулы:

$c \to \varnothing$ (1)

$bb \to ddd$ (2)

Проверим этот новый алгоритм на том же входном слове:

$$abbcabbca \xrightarrow{1} abbabbca \xrightarrow{1} abbabba \xrightarrow{2} adddabba \xrightarrow{2} adddaddda$$

Итак, НАМ сначала удалил все символы c и только затем заменил первое вхождение bb на ddd. Однако НАМ на этом не остановился и стал заменять остальные вхождения bb. Почему? Дело в том, что, пока применима хотя бы одна формула, НАМ продолжает свою работу. Мы должны принудительно остановить НАМ после того, как он заменил первое вхождение bb. Для этого и нужны заключительные формулы подстановки, после применения которых НАМ останавливается. Следовательно, в нашем алгоритме обычную формулу bb→ddd надо заменить на заключительную формулу bbaddd:

$c \to \varnothing$ (1)

$bb \mapsto ddd$ (2)

Вот теперь наш алгоритм будет работать правильно:

$$abbcabbca \xrightarrow{1} abbabbca \xrightarrow{1} abbabba \xrightarrow{2} adddabba$$

Слово, которое получилось после применения заключительной формулы (2), является выходным словом, т.е. результатом применения НАМ к заданному входному слову.

Проверим наш НАМ на входном слове, в которое не входит bb:

$$dcacb \xrightarrow{1} dacb \xrightarrow{1} dab$$

К последнему слову (dab) неприменима ни одна формула, поэтому, согласно определению НАМ, алгоритм останавливается и это слово объявляется выходным.

Пример 7.2 (*перестановка символов*). Пусть A={∅, a, b}. Преобразовать слово P так, чтобы в его начале оказались все символы a, а в конце – все символы b. Например: babba → aabbb

Решение Задача решается с помощью НАМ, содержащего всего одну формулу: ba→ ab

Пока в слове P справа хотя бы от одного символа b есть символ a, эта формула будет переносить a налево от этого b. Формула перестает работать, когда справа от b нет ни одного a, это и означает, что все a оказались слева от b. Например: babba → abbba → abbab → ababb → aabbb. Алгоритм остановился на последнем слове, т.к. к нему уже неприменима наша формула.

Задачи и упражнения для выполнения на занятии

1 Пусть слово P имеет следующий вид: $\underbrace{|\ldots|\ldots|}_{n} \otimes \underbrace{|\ldots||\ldots|}_{m}$, где ⊗ – один из знаков +, –, ×, /, ÷, ↑ или ↓, слева от которого указано n палочек, а справа – m палочек.

Реализовать соответствующую операцию в единичной системе счисления (в качестве ответа выдать слово, указанное справа от стрелки):

а) сложение: $\underbrace{|\,|\ldots|}_{n} + \underbrace{|\,|..|}_{m} \to \underbrace{|\,|\ldots|}_{n+m}$ (где n≥0, m≥0);

б) вычитание: $\underbrace{|\,|\ldots|}_{n} - \underbrace{|\,|..|}_{m} \to \underbrace{|\,|\ldots|}_{n-m}$ (где n≥m≥0);

в) умножение: $\underbrace{|\,|\ldots|}_{n} \times \underbrace{|\,|..|}_{m} \to \underbrace{|\,|\ldots|}_{n\times m}$ (где n≥0, m≥0);

г) деление нацело: $\underbrace{|\,|\ldots|}_{n} / \underbrace{|\,|..|}_{m} \to \underbrace{|\,|\ldots|}_{k}$ (где n≥0, m>0, k=n div m);

д) взятие остатка: $\underbrace{|\,|\ldots|}_{n} \div \underbrace{|\,|..|}_{m} \to \underbrace{|\,|\ldots|}_{k}$ (где n≥0, m>0, k=n mod m);

е) максимум: $\underbrace{|\,|\ldots|}_{n} \uparrow \underbrace{|\,|..|}_{m} \to \underbrace{|\,|\ldots|}_{k}$ (где n≥0, m≥0, k=max(n,m));

ж) минимум: $\underbrace{|\,|\ldots|}_{n} \downarrow \underbrace{|\,|\ldots|}_{m} \to \underbrace{|\,|\ldots|}_{k}$ (где n≥0, m≥0, k=min(n,m)).

Задачи и упражнения для выполнения дома

1 Определите НАМ, который уменьшает десятичное число на единицу.

2 Определите НАМ сложения по модулю 2 двух двоичных чисел.

3 Определите НАМ сложения двух двоичных чисел методом уменьшения одного числа на 1 и увеличением другого числа на 1 до тех пор, пока уменьшаемое число не станет равным 0.

4 Пусть A={∅, a, b}. Если в непустом слове P совпадают первый и последний символы, то удалить оба этих символа, а иначе слово не менять.

Лабораторная работа

Время выполнения – 2 часа.

Задания к лабораторной работе

1 Составить нормальный алгоритм Маркова для обработки слов.

2 Под «единичной» системой счисления понимается запись неотрицательного целого числа с помощью символа «|». Должно быть выписано столько палочек, какова величина числа; например: 2→ ||, 5 → |||||, 0 → ∅.

Вариант 1

1 Пусть A={∅, a, b, c}. Определить, входит ли первый символ непустого слова P ещё раз в это слово. Ответ: слово «да», если входит, иначе – «нет».

2 Пусть A={∅, |}. Считая слово P записью числа в единичной системе счисления, получить остаток от деления этого числа на 2, т.е. получить слово из одной палочки, если число нечётно, или пустое слово, если число чётно.

Вариант 2

1 Пусть A={a, b, c}. Определить, входит ли символ a в слово P. Ответ (выходное слово): слово a, если входит, или пустое слово, если не входит.

2 Пусть A={∅, 1, 2}. Считая слово P записью числа в троичной системе счисления, получить остаток от деления этого числа на 2, т.е. получить слово 1, если число нечётно, или слово 0, если число чётно. (Замечание: в чётном троичном числе должно быть чётное количество цифр 1.)

Вариант 3

1 Разработайте нормальный алгоритм Маркова, разбивающий пополам слово, состоящее из нечетного количества букв.

2 Определите нормальный алгоритм логического сложения и логического умножения двух двоичных чисел.

Вариант 4

1 Даны два слова, разделенные пробелом. Задайте нормальный алгоритм Маркова, удаляющий слово, состоящее из большего количества букв.

2 Определите нормальный алгоритм деления двух двоичных чисел с определением частного и остатка.

Вариант 5

1 Разработайте нормальный алгоритм Маркова, разбивающий пополам слово, состоящее из четного количества букв.

2 Определите нормальный алгоритм вычисления наименьшего общего кратного двух двоичных чисел.

Вариант 6

1 Составьте алгоритм Маркова, стирающий среднюю букву в слове.

2 Дано число, записанное последовательностью единиц. Составьте нормальный алгоритм Маркова, преобразующий число по формуле:

$$\varphi(n) = \begin{cases} n - 2, & \text{если } n > 5, \\ 1, & \text{если } n = 5, \\ 2 \cdot m, & \text{если } n < 5. \end{cases}$$

Вариант 7

1 Даны два слова, разделенные пробелом. Составьте нормальный алгоритм Маркова, проверяющий равно ли количество букв в слове.

2 Пусть A={∅, 0, 1}. Считая непустое слово P записью числа в двоичной системе, получить двоичное число, равное учетверённому числу P (например: 101 → 10100).

Вариант 8

1 Найти НОД двух чисел, записанных последовательностью единиц.

2 Пусть A={∅, 0, 1}. Считая непустое слово P записью числа в двоичной системе, получить двоичное число, равное неполному частному от деления числа P на 2 (например: 1011 → 101).

Вариант 9

1 Даны N слов, разделенных пробелами. Составьте нормальный алгоритм Маркова, определяющий количество слов.

2 Пусть A={∅, 0, 1, 2, 3}. Считая непустое слово P записью четверичного числа, проверить, чётно оно или нет. Ответ: слово 0, если чётно, и слово 1 иначе.

Вариант 10

1 Даны два слова, разделенных неизвестным количеством пробелов. Составить нормальный алгоритм Маркова, выполняющий склеивание этих слов.

2 Дана десятичная запись числа. Составить нормальный алгоритм Маркова, уменьшающий это число на 1.

Вариант 11

1 Составьте нормальный алгоритм Маркова, склеивающий произвольное количество слов, отделенных друг от друга одним пробелом.

2 Пусть A={∅, 0, 1, 2, 3}. Считая непустое слово P записью четверичного числа, получить остаток от деления этого числа на 4.

Вариант 12

1 Пусть слово P имеет чётную длину. Составьте нормальный алгоритм Маркова для удаления правой половины этого слова.

2 Пусть A={∅, 0, 1}. Считая непустое слово P записью двоичного числа, определить, является ли это число степенью 2 (1, 2, 4, …). Ответ: слово 1, если является, или слово 0 иначе.

Вариант 13

1 Дана последовательность из открывающихся и закрывающихся круглых скобок. Разработайте нормальный алгоритм Маркова, удаляющий парные скобки.

2 Пусть A={∅, |}. Считая слово P записью числа в единичной системе счисления, получить запись этого числа в троичной системе. (*Рекомендация*: следует в цикле удалять из «единичного» числа по палочке и каждый раз прибавлять 1 к троичному числу, которое вначале положить равным 0.)

Вариант 14

1 Пусть A={∅, a, b, c, d}. В непустом слове P переставить первый и последний символы.

2 Дано число в восьмеричной системе счисления. Разработайте нормальный алгоритм Маркова, увеличивающий это число на 1.

Вариант 15

1 Даны N слов, разделенных тремя пробелами. Составить нормальный алгоритм Маркова, удаляющий нечетные слова.

2 Пусть A={∅, 0, 1, 2}. Считая непустое слово P записью числа в троичной системе, получить запись этого числа в единичной системе.

Вариант 16

1 Дано слово, состоящее из букв {a, b, c, d}. Составьте нормальный алгоритм Маркова, подсчитывающий количество каждой буквы в слове.

2 Постройте нормальный алгоритм Маркова, реализующий вычитание двух целых чисел, представленных символами «1».

Вариант 17

1 Даны 2 числа M и N, разделенные знаком «?». Разработайте нормальный алгоритм Маркова, который вместо «?» ставит подходящий знак «<», «>», «=».

2 Даны 2 целых положительных числа: одно в троичной, другое в десятичной системах счисления. Разработайте нормальный алгоритм Маркова, позволяющий складывать эти числа в десятичной системе счисления.

Вариант 18

1 Пусть A={∅, a, b}. Определить, является ли слово P палиндромом (перевёртышем, симметричным словом). Ответ: слово a, если является, или пустое слово иначе.

2 Дано число в десятичной системе счисления. Составьте нормальный алгоритм Маркова, определяющий делится ли число на 2 без остатка. Если делиться, то левее написать «да», если не делится, то правее написать «нет».

Вариант 19

1 Дана последовательность одинаковых символов. Составить нормальный алгоритм Маркова, записывающий в 10-й системе счисления количество меток.

2 Дано число N, записанное последовательностью единиц. Составьте нормальный алгоритм Маркова, преобразующий число по формуле 2·N.

Вариант 20

1 Даны символы a, b, c, d, e в произвольном порядке. Составить нормальный алгоритм Маркова, который расположит числа в порядке возрастания.

2 Пусть A={∅, |}. Пусть слово P является записью числа 2^n (n=0, 1, 2, …) в единичной системе. Получить в этой же системе число n.

Вариант 21

1 Дано число в десятичной системе счисления. Разработайте нормальный алгоритм Маркова, умножающий это число на 2 и представляющий его в виде последовательности единиц.

2 Пусть A={∅, |}. Считая слово P записью числа n в единичной системе, получить в этой же системе число 2^n.

Вариант 22

1 Составить алгоритм Маркова, реализующий ограниченное вычитание.

2 Пусть A={∅, |}. Считая слово P записью числа в единичной системе, определить, является ли это число степенью 3 (1, 3, 9, 27, …). Ответ: пустое слово, если является, или слово из одной палочки иначе.

Лабораторная работа №9. ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМА

Краткие сведения

Характеристики сложности алгоритма

Временную сложность будем подсчитывать в исполняемых командах: количество арифметических операций, количество сравнений, пересылок (в зависимости от алгоритма). Временную сложность алгоритма α обозначим $T_\alpha(V)$. В качестве таких «элементарных» операций предлагается использовать следующие: присваивание: a := b; одномерная индексация a[i]; операторы выводы значения, оператор ввода значения; арифметические операции: *, /, -, +;

операции сравнения: a < b, a > b, a <= b, a >= b, a = b, a <> b; логические операции or, and, not, xor.

Ёмкостная сложность будет определяться количеством скалярных переменных, элементов массивов, элементов записей или просто количеством байт. Ёмкостную сложность алгоритма α обозначим $S_\alpha(V)$.

Параметр V, характеризующий данные, называют *объёмом данных* или *сложностью данных*.

Задачи и упражнения для выполнения на занятии

1 Пусть требуется построить МТ, которая прибавляет 1 к числу на ленте. *Замечание*. Входное слово состоит из цифр целого десятичного числа, записанных в последовательные ячейки на ленте. В начальный момент времени головка МТ находится напротив последней цифры числа.

Вычислите временную сложность и емкостную сложность машины Тьюринга.

2 Вычислите временную сложность и емкостную сложность машины Тьюринга для задач из соответствующей лабораторной работы.

3 Вычислите временную сложность и емкостную сложность машины Поста для алгоритма, который прибавляет 1 к числу на ленте. В начальный момент времени головка МП находится где-то над числом.

4 Опишите на языке блок-схем указанные ниже алгоритмы.

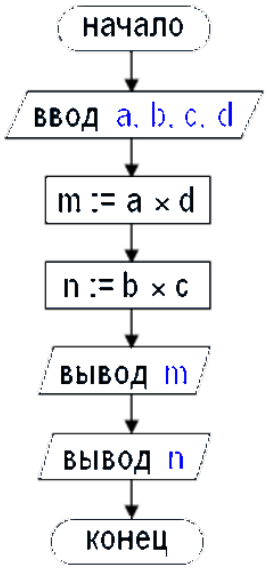
a) нахождения НОД (a, b) по алгоритму Евклида;

b) нахождения n-го числа Фибоначчи.

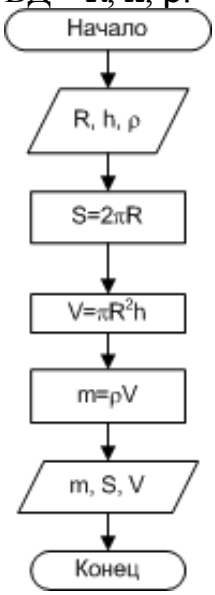
Вычислите сложность этих алгоритмов.

5 Вычислите сложность ($T_{min}\alpha$, $T_{max}\alpha$ и $T_{ср}\alpha$) алгоритмов, представленных в виде блок-схемы, в зависимости от объема указанных входных данных (ВД) (рисунок 9.1).

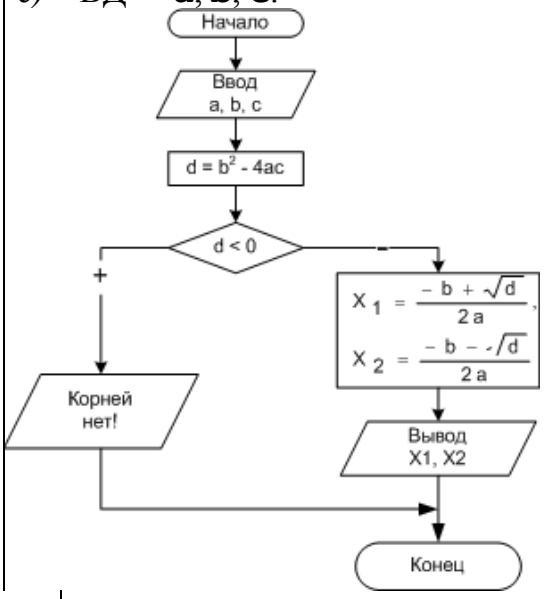
a) ВД – **a**, **b**, **c**, **d**.



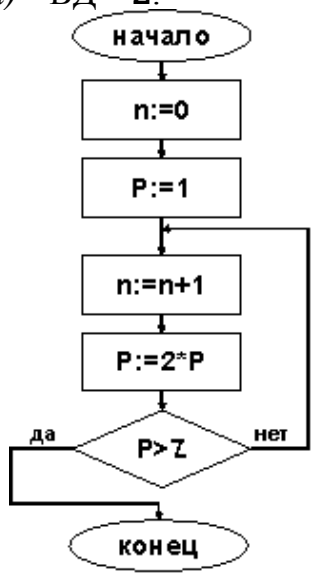
b) ВД – **R**, **h**, **ρ**.



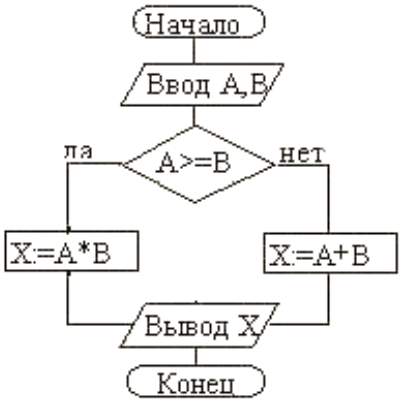
c) ВД – **a**, **b**, **c**.



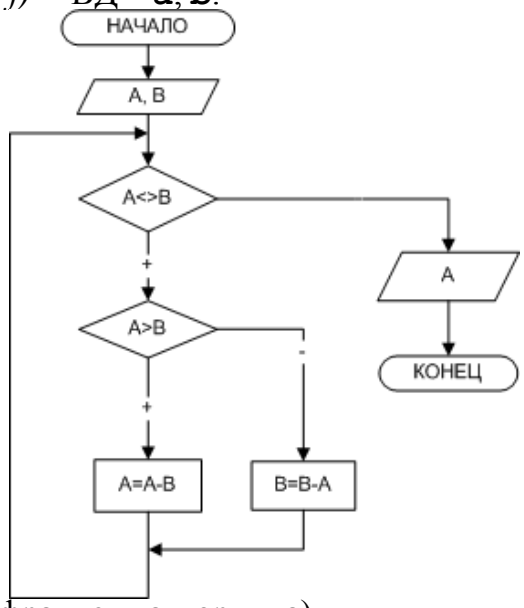
d) ВД – **z**.



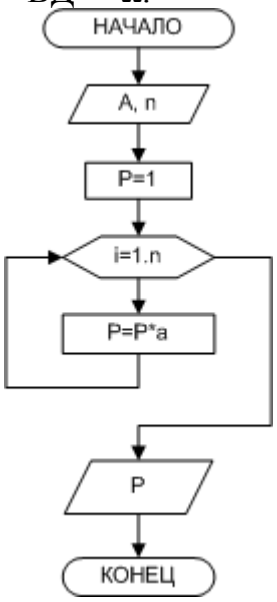
e) ВД – **a**, **b**.



f) ВД – **a**, **b**.



g) ВД – **n**.



h) ВД – **M**, **N** (приведен фрагмент алгоритма).

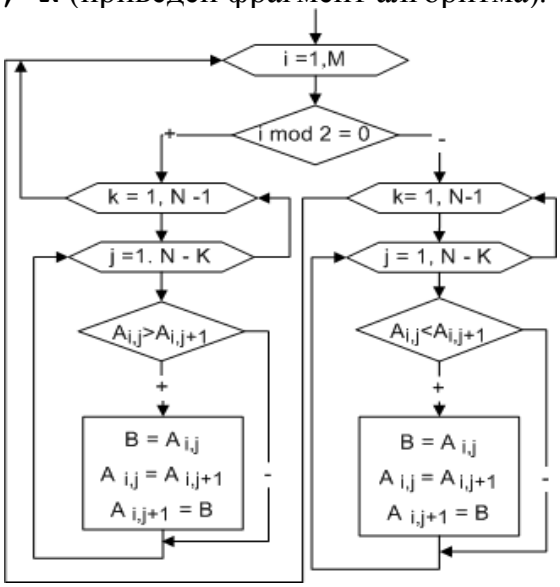


Рисунок 9.1 – Блок-схемы для задания 5

6 Вычислите сложность алгоритма метода сортировки одномерного массива:

a) сортировка методом выбора;

b) сортировка методом обмена, представленная на языке блок-схем (см. справа);

c) быстрая сортировка одномерного массива.

Вычислите $T_{min}\alpha$, $T_{max}\alpha$ и $(T_{cp}\alpha)$ сложности алгоритмов. Сравните сложности.

Задачи и упражнения для выполнения дома

1 На языке блок-схем опишите алгоритм:

a) вычисления *n!*; b) вычисления a^n;

c) вычисления *1!+2!+3!+…+n!*.

Постройте управляющий граф сложности алгоритма. Найдите сложность алгоритмов в зависимости от объёма исходных данных.

Рекомендуемая литература для подготовки к занятиям

1 *Верещагин* Н.К., *Шень* А. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. М. : МЦНМО, 1999.

2 *Гринченков* Д.В., *Потоцкий* С.И. Математическая логика и теория алгоритмов для программистов. М. : КноРус, 2010.

3 *Игошин* В.И. Математическая логика и теория алгоритмов. М. : Издательский центр «Академия», 2008. 448 с.

4 *Ильиных* А.П. Теория алгоритмов : учебное пособие. Екатеринбург, 2006. 149 с.

5 *Клини* С., *Весли* Р. Основания интуиционистской математики с точки зрения теории рекурсивных функций. М. : Наука, 1978. 272 с.

6 *Кнут* Д. Искусство программирования. 3-е изд. Т.1. Основные алгоритмы. М. : Вильямс, 2006. 720 с.

7 *Кнут* Д. Искусство программирования. 3-е изд. Т.2. Получисленные методы. М. : Вильямс, 2007. 832 с.

8 *Кнут* Д. Искусство программирования. 2-е изд. Т.3. Сортировка и поиск. М. : Вильямс, 2007. 824 с.

9 *Кнут* Д., Искусство программирования. М. : Вильямс, 2007. Т.4. Вып. 3. Генерация всех сочетаний и разбиений. 208 с.

10 *Кнут* Дональд, Искусство программирования. М. : Вильямс, 2007. Т.4. Вып. 4. Генерация всех деревьев. История комбинаторной генерации. 160 с.

11 *Колмогоров А. Н.* Теория информации и теория алгоритмов. М. : Наука, 1987.

12 *Кормен* Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. 2-е изд. М. : Вильямс, 2006. 1296 с.

13 *Марков* А.А., Нагорный Н.М. Теория алгорифмов. М. : ФАЗИС, 1996. 448 с.

14 *Миков А.И*. Информатика. Введение в компьютерные науки. Пермь, ПГУ. 1998.

15 *Могилев А. В*., *Пак Н. И*., *Хеннер Е.К*. Информатика : учебное пособие для студ. пед. вузов М. : ИЦ «Академия», 1999. 816 с.

16 *Могилев А. В., Пак Н. И., Хеннер Е.К*. Практикум по информатике : учебное пособие для студ. высш. учеб. заведений. М. : ИЦ «Академия», 2001. 608 с.

17 *Сайт* кафедры ИТ «Шаг за шагом». URL: http://it.kgsu.ru/.

18 Теория алгоритмов / Электронный учебник URL: http://ric.uni-altai.ru/Fundamental/teor-alg/

19 *Успенский* В.А. Машина Поста. М. : Наука, 1988.

Никифорова Татьяна Анатольевна

ТЕОРИЯ АЛГОРИТМОВ

Методические рекомендации
для проведения лабораторных работ
для студентов очной и заочной форм обучения направления 230700.62
«Прикладная информатика»

Редактор Е.А. Могутова

Подписано в печать20.01.14	Формат 60x84 1/16	Бумага тип. №1
Печать цифровая	Усл.п.л. 2,5	Уч.-изд.л. 2,5
Заказ № 7	Тираж 30	Не для продажи

РИЦ Курганского государственного университета.
640669, г. Курган, ул. Гоголя, 25.
Курганский государственный университет.